NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ESTABLISHING SEMANTIC INTEROPERABILITY, UNDER DENIED, DISCONNECTED, INTERMITTANT, AND LIMITED TELECOMMUNICATIONS CONDITIONS

by

Tebai Wissem

June 2014

Thesis Advisor: Dan Boger
Second Reader: Scot Miller

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis 06–20–2012 06–20–2014

4. TITLE AND SUBTITLE
ESTABLISHING SEMANTIC INTEROPERABILITY, UNDER DENIED, DISCONNECTED, INTERMITTANT, AND LIMITED TELECOMMUNICATIONS CONDITIONS

5. FUNDING NUMBERS

6. AUTHOR(S) Wissem Tebai

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A ____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	**12b. DISTRIBUTION CODE**
Approved for public release; distribution is unlimited	A

13. ABSTRACT (maximum 200 words)

The different approaches and technologies used for system integration and interoperability are explored in this thesis. Of particular interest is the use of the Data Distribution Service (DDS) open standard to integrate the component of any given command and control system working in a denied network environment. The method used for this research includes a review of past literature on the different specification to implement semantic interoperability for better and more efficient integration, as well as an exploration of the different functionalities and capabilities of DDS.

We present a middleware design based on the DDS specifications as developed by the Object Management Group. The design was influenced by the different limitations and requirements of the networking environment. Meanwhile, the proposed architecture also offers ways to implement semantic interoperability solutions in the system. Finally, the thesis describes a deployment scenario with a small network in order to accurately define the system controls that could impact the overall functionality of the DDS design, primarily through the Quality of Service (QoS) provisions.

14. SUBJECT TERMS Integration, interoperability, DDS, QoS, DIL, D-DIL, message-centric, data-centric, SOA	**15. NUMBER OF PAGES** 73
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ESTABLISHING SEMANTIC INTEROPERABILITY, UNDER DENIED, DISCONNECTED, INTERMITTANT, AND LIMITED TELECOMMUNICATIONS CONDITIONS

Wissem Tebai
First Lieutenant, Tunisian Air Force
B.S., Tunisian Air Force Academy, 2009

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(COMMAND, CONTROL, AND COMMUNICATIONS)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2014**

Author: Wissem Tebai

Approved by: Dan Boger
Thesis Advisor

Scot Miller
Second Reader

Dan Boger, Ph.D.
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The different approaches and technologies used for system integration and interoperability are explored in this thesis. Of particular interest is the use of the Data Distribution Service (DDS) open standard to integrate the component of any given command and control system working in a denied network environment. The method used for this research includes a review of past literature on the different specification to implement semantic interoperability for better and more efficient integration, as well as an exploration of the different functionalities and capabilities of DDS.

We present a middleware design based on the DDS specifications as developed by the Object Management Group. The design was influenced by the different limitations and requirements of the networking environment. Meanwhile, the proposed architecture also offers ways to implement semantic interoperability solutions in the system. Finally, the thesis describes a deployment scenario with a small network in order to accurately define the system controls that could impact the overall functionality of the DDS design, primarily through the Quality of Service (QoS) provisions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. MOTIVATION....2
B. PROBLEM STATEMENT....2
C. MAIN CONTRIBUTION....3
D. RELEVANCE TO DOD/DON....3
E. METHODOLOGY....3
F. ORGANIZATION OF THE THESIS....4

II. INTEROPERABILITY....5
A. LEVELS OF INTEROPERABILITY....5
1. Technical Interoperability....5
2. Syntactic Interoperability....6
3. Semantic Interoperability....7
B. INTEGRATION APPROACHES....8
1. Point-to-Point Integration....8
2. Service-Oriented Architecture....8
3. Decentralized Service-Oriented Architecture....9
C. INTEROPERABILITY AND DATA MODELING....10
D. IMPLEMENTING INTEROPERABILITY....11
1. OMG Specifications and Activities....11
2. World Wide Web Consortium (W3C) Standards....12
E. SUMMARY....14

III. DATA DISTRIBUTION SERVICES....15
A. THE DDS STANDARD....15
B. ADVANTAGES OF THE DDS....17
C. SEMANTIC INTEROPERABILITY THROUGH DDS....18
D. ADOPTION....19
E. MIDDLEWARE IN COMMAND AND CONTROL SYSTEMS....20
F. SUMMARY....21

IV. ARCHITECTURE DESIGN....23
A. DDS DESIGN AND DISADVANTAGED NETWORKS....23
1. Data Compactness....23
2. Network Recovery....24
3. Limited Resource....24
4. Graceful Degradation....25
B. DDS DESIGN AND SEMANTIC INTEROPERABILITY....25
C. PROPOSED ARCHITECTURE....26
1. The Application Layer....27
2. The DCPS API....27
3. The RTPS Engine....28
4. The RTPS Wire Protocol....28
5. The Controller....28

6. The Discovery Protocol....29
D. DEPLOYMENT SCENARIO....29
1. Architecture of the Distributed System....29
2. QoS Settings....30
a. Reliability....31
b. Durability....32
c. History....33
d. Ownership....34
e. OwnershipStrength....35
f. ContentFilteredTopic....35
E. CONCLUSION....37
V. CONCLUSION AND RECOMMENDATIONS....39
A. SUMMARY....39
B. RECOMMENDATIONS AND FUTURE WORK....39
APPENDIX. DATA MODELING IN DDS....41
LIST OF REFERENCES....51
INITIAL DISTRIBUTION LIST....55

LIST OF FIGURES

Figure 1. Levels of interoperability (from [1]).....5
Figure 2. Point-to-Point Integration. 8
Figure 3. Service-Oriented Architecture. 9
Figure 4. Decentralized SOA 10
Figure 5. DDS-PS architecture (from [20])....16
Figure 6. The Publish-Subscribe service paradigm (from [20])....17
Figure 7. DDS design for D-DIL networks....27
Figure 8. Architecture of the distributed system 30
Figure 9. Durability QoS Policy (from [29])....32
Figure 10. History QoS Policy (from [29]) 33
Figure 11. Ownership QoS Policy (from [29])....34
Figure 12. OwnershipStrength QoS Policy (from [29]) 35

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Comparison of web services and DDS (from [26]) 20
Table 2. QoS policies offered by DDS 31
Table 3. QoS settings of the distributed system 37

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface
C2	Command and Control
CDR	Common Data Representation
CIM	Computation Independent Model
DCPS	Data Centric Publish Subscribe
DDS	Data Distribution Services
DISA	Defense Information System Agency
DOD	Department of Defense
D-DIL	Denied-Disconnected Intermittent and Limited
IDL	Interface Definition Language
EXI	Efficient XML Interchange
IT	Information Technology
LOD	Linked Open Data
MDA	Model Driven Architecture
MDMI	Model Driven Message Interoperability
MOF	Meta Object Facility
OC	Operation Center
OMG	Object Management Group
OWL	Web Ontology Language
PIM	Platform Independent Model
PS	Publish Subscribe
PSM	Platform Specific Model
QoS	Quality of Services
RDF	Resource Description Framework
RDBMS	Relational Data Base Management System
RTPS	Real Time Publish Subscribe
SIOp	Semantic Inter-Operability
SOA	Service Oriented Architecture
SSDS	Ship Self Defense System
UML	Unified Modeling Language
UUA	Unmanned Underwater Vehicles
XML	Extensible Markup Language
XSD	XML Schema Definition

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I thank Allah, the almighty, for his blessings and help to achieve this work and carry my journey of life with ease and success.

I gratefully acknowledge my advisor Dr. Dan Boger for his guidance and thoughtful insights throughout the course of this thesis. Thanks go to my second reader Mr. Scot Miller for his time, advice, and assistance.

I would like to dedicate this work to my family and friends and express my gratitude and my love to my dear parents who have always been there for me and gave me a beautiful model of hard work and perseverance.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The volume of transmitted and processed data in current information technology (IT) systems is growing dramatically, which makes for huge challenges managing this mass of data, especially when that requires various applications and systems to interact. Among these challenges is the ability to process data from their source in a meaningful manner. In other words, the semantic meaning of the exchanged data must be preserved and understandable. Another important aspect of this task is to achieve interoperability between different systems and applications when the need for their integration is essential. This integration is usually complex and expensive. Moreover, it does not necessarily guarantee the exchange of the information's semantics.

Among the entities interested in this issue, military IT and Command and Control systems are involved in a number of relevant projects. As these systems are now more advanced and centered on the user, researchers have reached a point where the system can begin to take over initial decision-making and derive knowledge and provide recommendations for the user. The sharing of information among disparate systems requires integration at the semantic level for systems to communicate information efficiently. This integration step is called semantic interoperability (SIOp), and while critical, it is not fully understood even with advanced capabilities.

This thesis assesses the Object Management Group (OMG) efforts on Data Distribution Services (DDS) for its applicability to military IT/C2 systems operating in a denied environment. DDS is advertised to provide current solutions for semantic interoperability issues both in commercial and military distributed systems. Our first goal is to explain how the specifications of the DDS are applied in real-world systems and how they contribute to SIOp.

The other facet of this research is to propose architecture for the Navy IT/C2 systems based on DDS specifications in order to achieve SIOp. This architecture could be an example to follow when implementing SIOp in Department of Defense (DOD) IT/C2

systems in a Denied, Disconnected, Intermittent, and Limited (D-DIL) environment. Finally, substantive recommendations on costs, benefits, and risks will be addressed.

A. MOTIVATION

Due to the advances in today's technology, information systems are effective at dealing with data infrastructure and handling multiple data formats, as well. However, they are not very good at understanding how the semantics of data in independent data sources are related, which affects the ability to transfer the information accurately. Too often, the way each system interprets data is implicit in the specific design and operation of the system. The presence of many differences in structure, terminology, viewpoint, and notation makes those systems hard to integrate, negatively impacting their capability to inter-operate.

Many of the issues related to semantic interoperability in IT/C2 systems are also relevant to other communities, such as semantic web, artificial intelligence, knowledge representation, ontology, and digital libraries. Our focus will be the use of DDS as a very interesting middleware to achieve interoperability. This middleware is already implemented in both commercial and DOD systems.

Naval IT/C2 systems operating in D-DIL networks are always complex and hard to integrate, and SIOp is one of the major problems in that process. It is clearly beneficial to design an architecture that assists those systems to operate effectively and implement semantic interoperability. One of the design solutions here is the use of open standards such as DDS, which provides interoperability and assures the sharing of semantic information in distributed systems.

B. PROBLEM STATEMENT

Information sharing is essential for an integrated approach to defense systems, counter-terrorism activities, business and government intelligence, and collaboration and integration of enterprise applications. However, this fundamental capability of information sharing has remained expensive and difficult to achieve in information systems that are usually isolated, stove piped, and customized. The inability of systems to

achieve semantic interoperability hampers the ability of defense organizations to work together in terms of processes, services, and information resources. Thus, the use of open-standard solutions presents great potential to interoperate our systems effectively. A worthy goal is to reduce the cost of integration, introduce the SIOp and achieve better decision making through sharing semantic information with the use of open standards like DDS.

C. MAIN CONTRIBUTION

This research investigates the application of DDS in the denied environment with SIOp emphasis. DDS architecture will be proposed by applying selected specifications from the Object Management Group (OMG) standards in a D-DIL environment. Further study will be conducted to provide recommendations on costs, benefits, and risks.

D. RELEVANCE TO DOD/DON

DDS has been adopted and mandated by many military and civilian organizations. Naval combat systems in the U.S. and worldwide are using DDS. It has been implemented into a number of defense systems, such as Ships Self Defense Systems (SSDS) and Aegis. The goal of this research is to explain how SIOp is implemented through DDS on some of those systems, and how it can be improved by introducing more of the DDS specifications. The proposed architecture suggests how this OMG standard can be beneficial for systems integration without losing the semantics of the shared data.

E. METHODOLOGY

This research contains a study of real-world IT/C2 systems using DDS with SIOp implementation. A prototype will be suggested based on both the specifications provided by the OMG group and the observations made of real systems integrating DDS.

F. ORGANIZATION OF THE THESIS

This thesis consists of the following chapters:

Chapter II will present background information and a condensed literature review of the different methods for SIOp implementation and, in particular, of the different specifications described by the OMG group and other groups.

The open standard DDS is detailed in Chapter III to explain how SIOp is supported within the different specifications. An overview is given of the state-of-the-art applications of the DDS in commercial and military IT/C2 systems.

Chapter IV introduces a proposed architecture that helps to integrate defense systems or sub-systems working in a D-DIL environment with a focus on establishing and improving SIOp.

Finally, a summary of the research results and recommendations for further research are presented in Chapter V.

II. INTEROPERABILITY

Current information systems are composed of many standards and tools. When it comes to connecting two or more different legacy systems, the issue of interoperability comes to the surface making the integration a complex and expensive task. In this, chapter, we will define several interoperability levels and describe the different approaches and specifications used to implement interoperability in general and SIOp in particular.

A. LEVELS OF INTEROPERABILITY

Interoperability is the ability for systems, units, or forces to provide services to and accept services from other systems, units, or forces, and to use the services so exchanged effectively together [1]. There are six levels of interoperability presented in Figure 1; however, only the bottom three will be addressed.

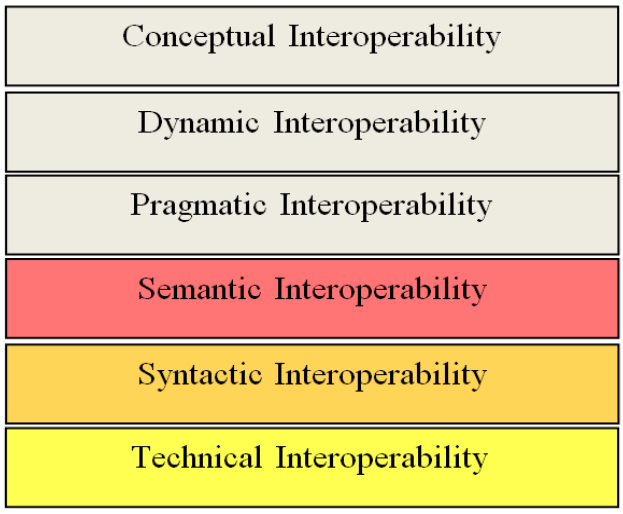


Figure 1. Levels of interoperability (from [1]).

1. Technical Interoperability

Technical interoperability requires communication infrastructure to be established and results in the exchange of bits and bytes. There is no common language or data definition. Technical interoperability is simply the ability to send signals or bytes through

a reliable physical connection. Technical integration is more feasible and less complex than other interoperability levels, due to the hardware standardization and the large number of integration solutions on the market.

2. Syntactic Interoperability

The syntactic interoperability specifies a common data structure to be used. Its focus is integrating systems via a set of standard data formats. These standards permit the specification of a number of syntactic rules to be obeyed [1]. The rules are validated by the implementation of a sentence recognizer in every used language.

Interoperability at the syntactic level provides systems with the ability to clearly exchange the structure of data and establish a common communication infrastructure. Syntactic specifications are useful for defining the form of expressions, sentences, and data objects. However, they cannot verify whether the content is semantically correct or not.

The use of a certain data structure provides more knowledge about the data transmitted than simply having an ambiguous flow of bits, as can occur at the technical level [1]. Syntactic interoperability enables the exchange of well-formed expressions with syntactic error detection. This can be a source of confusion and ambiguity when it comes to data interpretation because an expression can be syntactically valid, but its meaning is completely wrong. Therefore, the meaning of the exchanged expression is left to be interpreted by the communication endpoints.

A good and simple example of having syntactic interoperability only is the data delimiter mentioned in [1]. It is an agreed upon sequence of characters that are used to specify limits between data elements, which do not attribute any semantics to the expressions. In the Extensible Markup Language (XML), < > is used as a delimiter. It can tell us two things: there is a message where the content starts after "<" and ends before ">," and there are different types of content in the message. The syntax provided in the XML standard gives a user the ability to differentiate the structuring of his information. Thus, the user can specify the different types of data built within the messages without conveying the actual meaning of those messages.

3. Semantic Interoperability

Semantic interoperability was defined in [2] as the ability of different information systems to communicate information consistent with the intended meaning of the encoded information. This involves both the processing and presentation of information so that it conforms to the intended meaning regardless of the source of information. Systems are semantically interoperable when they can exchange the meaning of data in an unambiguous manner, achieve a common understanding, and detect when something may be semantically wrong.

In order to achieve semantic interoperability, the use of common information models is needed [3], [1]. Those data models should include the meaning of the information agreed upon, known, and used by the different participants. The example in [3] shows how semantic interoperability can be achieved using the calendar model instead of emails. The email application does not understand the message content (i.e., information about scheduling) and then decide how to manage the message, but using the calendar application in this scenario provides better information sharing between the user and the calendar process.

Calendar Process #1:

1. Email: “Meeting Monday at 10:00.”
2. Email: “Meeting moved to Tuesday.”
3. Email: “Here’s dial-in info for meeting…”
4. You: “Where do I have to be? When?”
5. You: (sifting through email…)

Calendar Process #2:

1. Calendar: (add meeting Monday at 10:00)
2. Calendar: (move meeting to Tuesday)
3. Calendar: (add dial-in info)
4. You: “Where do I have to be? When?”
5. You: (check calendar)

B. INTEGRATION APPROACHES

Interoperability and integration are similar terms when it comes to describing how applications or systems need to reliably communicate meaningful information. Following are three different integration approaches.

1. Point-to-Point Integration

Point-to-point integration, shown in Figure 2, is used by traditional technical management. Integration logic is introduced between all connected nodes, and the data or object states are maintained in each application [3]. Traditional middleware, such as messages, queues, and sockets, are used to convey information and handle the different interactions. For a small system, this is manageable. However, as systems grow, each new component may have many interactions. If n is the number of nodes, the number of interactions would be on the order of n squared, which grows exponentially.

A large number of interactions are hard to control, and understanding all those interactions is difficult, especially across various suppliers [4]. This model is also costly to maintain and it is not easy to upgrade or reuse.

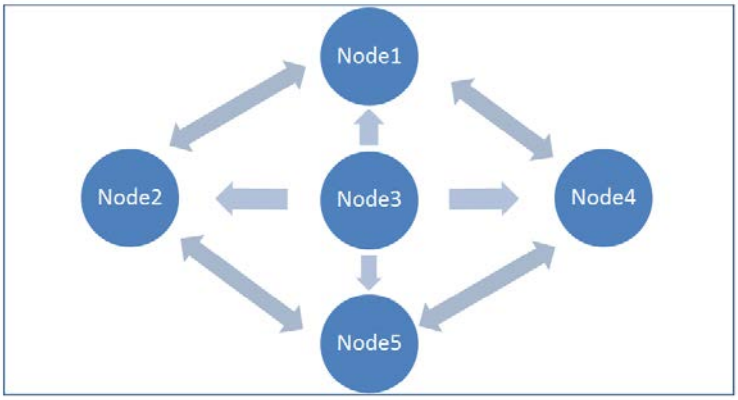


Figure 2. Point-to-Point Integration.

2. Service-Oriented Architecture

The Service-Oriented Architecture (SOA) presented in Figure 3 is the current enterprise service-centric integration approach. It is the most common approach for

integrating different systems and applications [3]. The state and data are maintained in servers and databases, and information is delivered to clients upon request. Interoperability is provided using traditional entities like Brokers and Relational Data Base Management Systems (RDBMS).

This approach presents much lower costs for integration, maintenance and upgrades, due to the fact that applications are loosely coupled and reuse is easier. However, this architecture has a single point of failure because both data and states are usually kept in the server. Moreover, SOAs work best in an intranet environment where everything is well connected and the business rules are well established. This is generally not the case for DOD.

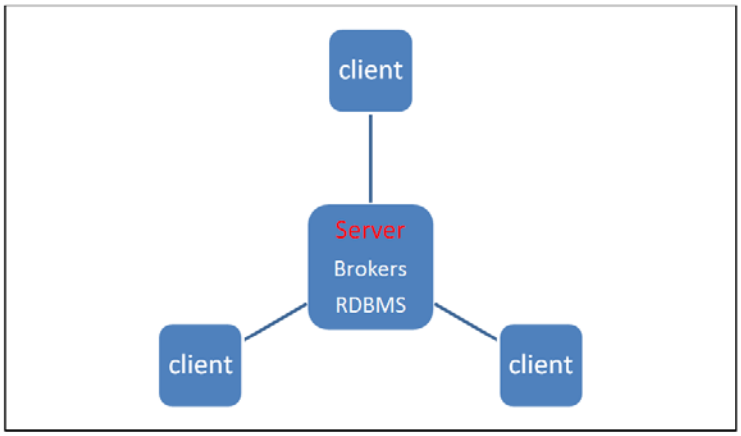


Figure 3. Service-Oriented Architecture.

3. Decentralized Service-Oriented Architecture

The approach presented in Figure 4 employs a data-centric integration model to decouple applications and systems [4], also known as the Data-Bus architecture. It presents a flexible, point-to-point deployment with the same SOA benefits (Discoverable, Common Interfaces). It is also more reliable since there is no single point of failure (State is maintained in the data-bus) [3]. Integration is much simpler since components can be added, upgraded, or replaced as a routine part of a program's evolution.

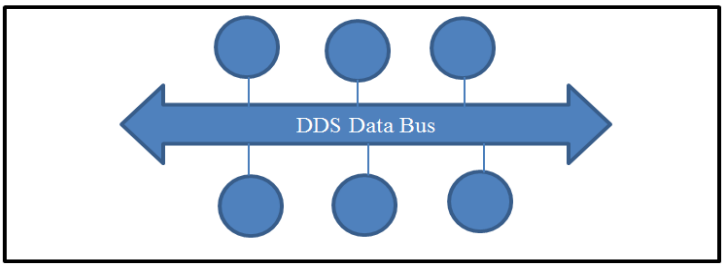


Figure 4. Decentralized SOA

C. INTEROPERABILITY AND DATA MODELING

Data modeling is the fundamental tool for building the syntax and semantics of data so that information can be exchanged without any interoperability issues. The following paragraph provides a detailed explanation of the data-modeling notion.

A model is a representation of an entities, functions, or behavior of an application or a system. A formal model is based on one or many languages that define specific sentences, which consists of syntax and attribute meanings (semantics), relationships, and rules of analysis, and inference of those sentences [5]. The syntactic representation could be textual or graphical. However, the semantics are determined by text-based definitions, deduction from the model statements, or by translating higher-level language structures into other structures that have a well-defined meaning. This is similar to relational database modeling where a diagram with boxes and lines and arrows is representing the system. That diagram is usually supported by a definition of the meaning of a box (entity), the meaning of a line (relationship), or the meaning an arrow (constraints or description of a relationship between entities).

As defined in [3], a data model is the representation that describes the data about the elements that exist in your domain. Once the data is discovered, many modeling standards can be used to provide a consistent model that defines data specifications, such as representation, semantics, and constraints. In the DDS case, applications communicate by exchanging discoverable data objects typically described by standard data models like Extensible Markup Language (XML) and Unified Modeling Language (UML).

D. IMPLEMENTING INTEROPERABILITY

This section will explore many of the standards used to help with SIOp issues. These standards are implemented in different technologies, such as middleware and semantic web, and can be related either to architectures, interfaces, or modeling standards for system interoperability.

1. OMG Specifications and Activities

The OMG is a software-standards body responsible for developing technologies like modeling languages, middleware, and many other software standards. As one of the largest computer industry consortiums with hundreds of member organizations including IBM, Microsoft, Oracle, Red Hat, and Tibco [6], OMG has adopted many standards such as DDS and CORBA to support system integration. Those standards are used to support communication across different platforms and to solve interoperability issues, using many of the following OMG specifications.

Model Driven Architecture (MDA: The OMG MDA Guide v1.0.1 defines MDA as an approach "separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform" [5]. This separation is done by generating different models for any given system. Those models include specifications for the Platform-Independent Model (PIM), Platform-Specific Model (PSM) and a number of interface definitions that describe the implementation of the base model on different middleware platforms enabling the interoperability both within and across platform boundaries [5]. Therefore, the MDA goal is to enable different applications to be integrated by explicitly relating their individual models. The different MDA models are more detailed according to the following:

- Computation Independent Models (CIMs) model the environment in which the system will operate.
- PIMs model the operation of the system independently of its implementation platform.
- PSMs model the system in the context of a specific platform choice.
-

Meta Object Facility (MOF): The OMG has developed MOF as the standard, abstract language used for metamodels specifications. When having different metamodels and defining different domains, MOF is used to generate codes that enable those models to interoperate through metadata management [7], [8]. Therefore, MOF typically offers an open-standard and platform-independent framework for metadata management and interoperability.

Unified Modeling Language (UML): UML is the standard language most often used by the OMG for modeling discrete systems. It provides a visual diagram that presents the architectural blueprint of a given system [9]. The diagram typically includes elements such as components, activities, interactions, and interfaces. It is upgraded by including Action Semantics (AS) and the Object Constraint Language (OCL) [10]. The latter provides a language for specifying constraints within UML models. The Foundational UML (fUML), a subset of UML, has been presented and specified with precise semantics for class modeling using some common logic [11].

Model Driven Message Interoperability (MDMI): The goal of MDMI is "to provide a declarative, model-driven mechanism to perform message data transformation" [12] for the financial services domain. A message data transformation is to be carried out in two steps. First, a message syntax model is used to define how values are extracted from a source message and identified as semantic elements, which are "the smallest semantic entities contained in a message format" [12]. Second, source semantic elements are mapped to business elements from a central Domain Dictionary. Those elements can then be used to derive equivalent semantic elements for the target message format.

2. World Wide Web Consortium (W3C) Standards

The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the web. Since the introduction of semantic web standards by Tim-Berners Lee over ten years ago, there has been growing interest and application of the W3C standards to provide web-scale

semantic data exchange, federation, and inference capabilities [13]. The following are some of the W3C specifications that support SIOp within the semantic web.

Web Ontology Language (OWL): OWL is the ontology language designed for semantic integration across the web. In information domains, the terms used to represent and describe an area of knowledge are called “ontologies,” which are used by databases and applications for information sharing [14]. OWL is used by different applications to process the web content in order to achieve better machine interoperability in the web. This standard uses three sub-languages (Lite, DL, and Full) and provides formal semantics and extra vocabulary for SIOp [5].

XML Schema Definition (XSD): XSD is used to describe a set of rules to which XML documents must comply for validation purposes. The rules are about the structure and the constraints of the XML documents’ content [15]. XSD determines the attributes and elements that can exist in an XML document and the data types this attributes and elements can have. Then, it verifies that each of them complies with the description given by the rule sets.

Linked Open Data (LOD): LOD provides advanced techniques for information publication to make data easily accessible and linked across the web [16], [17]. LOD was designed based on web standards like HTTP and URL for large scale web integration. LOD aims to link the data available on the web so persons and machines can explore and query data more efficiently. LOD describes practices to explore, share, and connect knowledge and data on the semantic web.

Resource Description Framework (RDF): The RDF is used to define the semantics for data files based on XML. RDF is not a language but a data model standard that relates to interoperability [18]. The model is composed of nodes that represent web resources. Those nodes are connected by arcs representing the properties of the resources. RDF is used along with a schema for more semantic specifications.

E. SUMMARY

In this chapter, the different levels of the interoperability have been described. Several activities and standards for semantic implementation in the data modeling have been presented. Most of those standards and tools are presented by the OMG and the W3C groups. With this information as background, the implementation of these standards to support SIOp using DDS will be investigated in the Chapter III.

III. DATA DISTRIBUTION SERVICES

Many of the issues relating to semantic interoperability in IT/C2 systems are also relevant to other communities. Therefore, it is valuable to look at the work of other communities in the area of semantic interoperability. These include semantic web, artificial intelligence, knowledge representation, ontology, and digital libraries [2]. The primary focus is about the architectural level of data modeling; thus, we will be looking at some of the work done by the Object Management Group developing DDS as a very interesting middleware used to integrate systems and assure their interoperability.

A. THE DDS STANDARD

The DDS standard was created and is managed by the OMG group. DDS is the first open international standard using the Publish Subscribe architecture (PS) for real-time systems. DDS has many important features. For example, it allows a fine control of real-time Quality of Service (QoS) parameters, such as bandwidth control, delivery deadlines, reliability, and resource limits [4].

OMG-DDS is essentially designed around the data-centricity concept in order to meet the challenges of real-time systems [19]. The PS architecture is the core of DDS. It enables entities called publishers to send data and other entities called subscribers to get it immediately. This is different from the server/client architecture because no servers are required and data is sent directly from the source.

PS architecture is shown in Figure 5, where the DDS provides a registration process through Data Writers and Data Readers included in each entity. This allows nodes or topics to join and leave at any time with the ability to specify the QoS requirements for every publisher and subscriber.

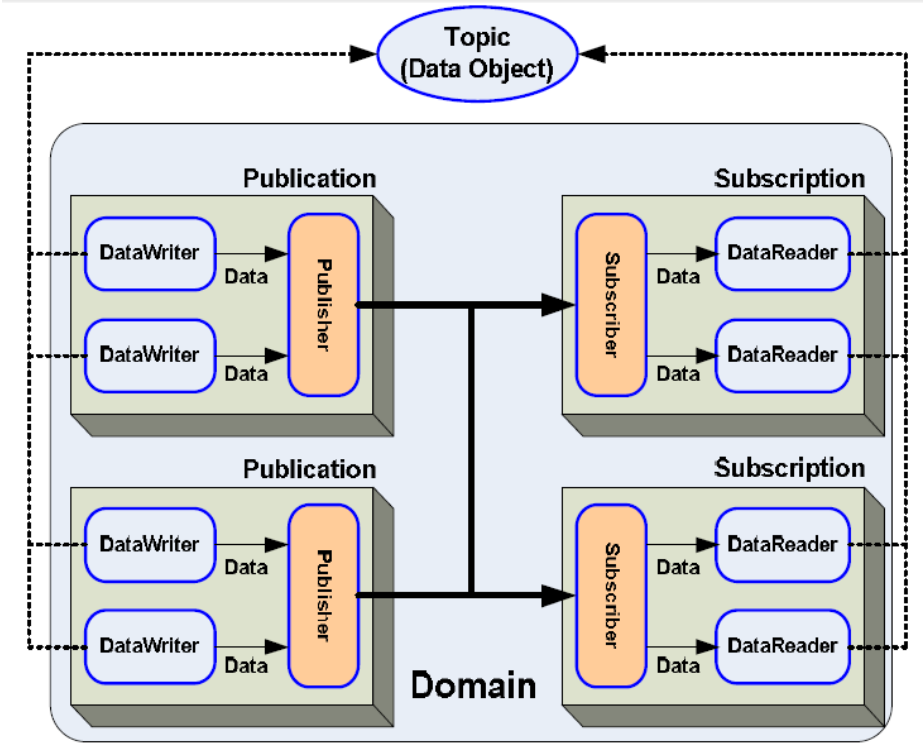


Figure 5. DDS-PS architecture (from [20])

The PS model connects anonymous information publishers with information subscribers [4]. The PS service paradigm shown in Figure 6 is described as distributed applications composed of processes running in a separate address space, and possibly on different hosts. Those processes (participants) may publish or subscribe to typed data-streams identified as topics. These topics are data objects that represents information that each node publish and update or subscribe to get updates, and each of these topics is identified by a unique name. Publishers and subscribers are decoupled in space, time, and flow. In other words, the network nodes can be anywhere, nodes can join or leave in any order at any time, and data delivery may be "best effort," reliable, or at controlled bandwidth which proves that DDS can be used flexibly in different systems. The PS paradigm shown in Figure 6 presents many features, such as asynchronous and anonymous publishers and subscribers, decoupled applications, and multicasting.

Topics may have many data channels identified by "keys," allowing nodes to subscribe to possibly thousands of similar data streams with a single subscription,

increasing the DDS scalability. When receiving the data, the middleware can use the key to sort it and deliver it for efficient processing [4].

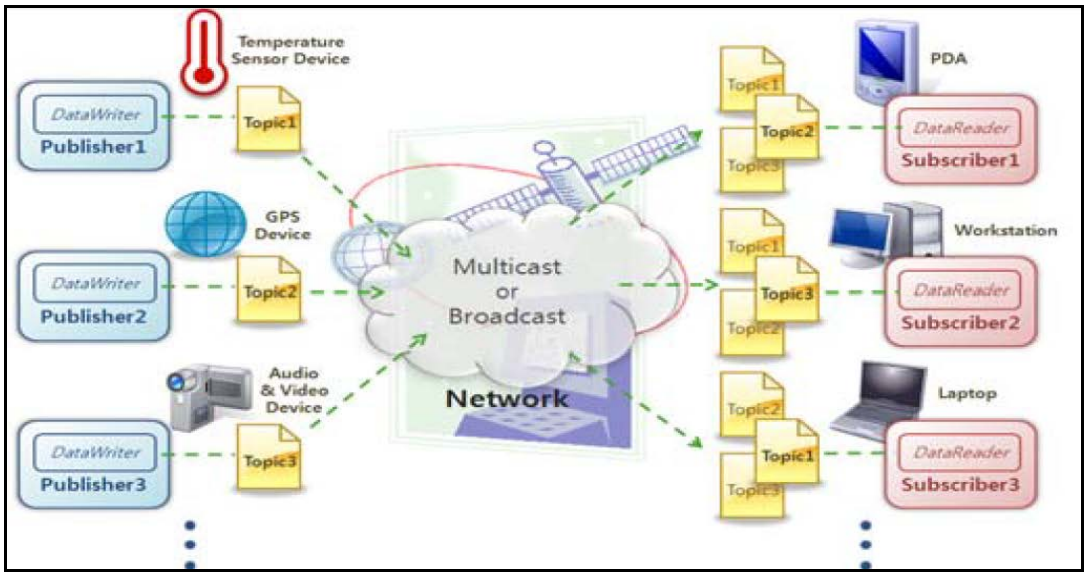


Figure 6. The Publish-Subscribe service paradigm (from [20])

A rich set of QoS is provided by DDS to specify the communication parameters between participants, properties of the overall model and the topics themselves. The QoS parameters such as “Durability” and “Ownership” enable the control and configuration of the local and end-to-end properties of DDS entities to meet the application requirements [21].

DDS is fundamentally designed to work over unreliable transports, such as UDP or wireless networks [6]. Its design does not require central servers or special nodes, and can be efficient for direct, peer-to-peer communication, or even multicasting, which favors its deployment in real-time distributed systems [22].

B. ADVANTAGES OF THE DDS

DDS has been improving since the time it was first introduced in 2001. DDS applications have been built and implemented in a wide range of real-world systems thanks to many qualities and attributes described by the following:

- **Open Interfaces**: DDS supports the development of modular, loosely coupled, and open-architecture information systems. It supports standard interfaces between different components and favors the elimination of stovepipes, closed and proprietary architectures which reduce integration complexity, maintenance and upgrade costs. DDS presents an open standard Application Programming Interface (API) for publish-subscribe messaging that reduces integration logic, and scales to systems of systems.
- **Performance**: DDS can be integrated efficiently in many different systems from small- embedded to large-distributed ones. DDS can meet the performance requirements for small enterprises to complex real-time systems [23] including throughput, tight latency, and bandwidth. In fact, the DDS performance has been demonstrated in many domains, such as traffic monitoring, command and control systems, combat systems, and financial data-distribution.
- **QoS**: DDS implementation includes a wide-ranging set of QoS policies. These offer control over the communication and data exchange behavior more than any other middleware [23]. QoS parameters allow integration of systems with modules that require differing update rates, reliability, and bandwidth control [6] which provides better interoperability and better flexibility.
- **Many implementations**: The DDS standard has become more available for use since many middleware implementations support the DDS API or Real Time Publish Subscribe (RTPS). More than thirty companies have adopted and supported the original specification, including Ericsson, IBM, MITRE, Nokia, Oracle, RTI, and THALES. Many military organizations have played a significant supporting role as well (see section D).

C. SEMANTIC INTEROPERABILITY THROUGH DDS

The SIOp in DDS is guaranteed through applications that communicate by exchanging discoverable data objects usually described by a common data model. The data model is usually defined using one or more semantic specifications, such as the OMG Interface Definition Language (IDL), XSD, XML, or programmatic interfaces [23]. The data model can also be created from a Unified Modeling Language (UML) model. These specifications are standards for semantic modeling mostly developed by the OMG and W3C groups and introduced in Chapter II. The DDS also relies on the MDA architecture to interlink the different platforms or subsystems with the support of meta-modeling specifications like MOF.

The protocols and interfaces that are supported by DDS enable this middleware to be semantically interoperable with most existing standards. The DDS-RTPS wire protocol provides seamless interoperability across platforms and programming languages. The programming languages are supported within the DDS design from the beginning and favors this middleware as the only standard messaging API for C and C++, which also supports Java, Ada, JMS, C# and HTTP interfaces [23]. The DDS-RTPS also provides interoperability among the different DDS implementations [24], and most vendors have supported this protocol and demonstrated its efficient usage for resolving interoperability problems.

D. ADOPTION

Many military and civilian organizations have adopted or mandated DDS around the world [24]. A number of naval combat systems in the U.S. and worldwide have been using DDS, such as the Aegis, Ship Self Defense System (SSDS) [25] and DDG 1000 programs. It is deployed by allied navies, including those of Japan, Germany, and the Netherlands. The following are some of the military organizations and systems that have adopted DDS:

- U.S. Navy: The Naval Sea Systems Command Dahlgren Division has developed a high-performance, radar and ship control system with DDS. This system has evolved into the basis for the Navy Open Architecture standard [6].
- Air Force, Navy: Net-centric Enterprise Solutions for Interoperability [24].
- Unmanned Underwater Vehicles: Bluefin Robotics produces various Unmanned Underwater Vehicles (UUVs) with DDS technology [6].
- Defense Information Systems Agency (DISA): mandated the standard within the DOD Information Technology Standards and Profile Registry (DISR) [24].
- UK Ministry of Defense: Generic Vehicle Architecture, an interoperable open architecture for unmanned vehicles [24].

DDS is also used commercially in a number of industries, including communications, financial services, transportation, industrial automation, air traffic control, mobile asset tracking, and medicine. Among the civilian organization users, we find NASA Robotics and the car manufacturer Volkswagen. Finally, a number of

universities worldwide are using DDS in active research projects, including MIT and ETH Zurich.

E. MIDDLEWARE IN COMMAND AND CONTROL SYSTEMS

In this section, we present a comparative analysis described in [26] that evaluates the performance of DDS and web services in C4I systems. Among the web services, there is the semantic web described in Chapter II. The test environment was designed based on C4I requirements of a weapon system, i.e., the data exchange needs to be real-time, the communications must be carried even in limited bandwidth, and the system must be distributed with no single point of failure. Latency measurements for applications were conducted in order to calculate the efficiency of the two middleware types in discovering nodes in the network and maintaining interactions between different entities. Other measurements were done as well for QoS and throughput. The results are summarized in Table 1.

	Web Services	**DDS**
Latency	Larger delay in this request-response messaging format	Negligible delay due to global data space and publish/subscribe mechanism
QoS	Further development is required to ensure right QoS capability	Matured QoS control for Topic, Publishers and Subscribers
throughput	Performance degrades with increase in data size	Performance is quite stable and degradation is for unicast implementation.
Framework Type	Message Centric Request-Response	Data-Centric Publish-Subscribe
Bandwidth Requirements	Large overhead due to XML documents	Negligible due to shared memory and global data space concept
Real-Time Support	Not yet realized completely	Matured and realized in various defense establishments.

Table 1. Comparison of web services and DDS (from [26])

The results show that the DDS presents better performance control and flexibility compared to web services when implemented in C4I systems. When using DDS for D-DIL networks, there are more issues that need to be addressed.

F. SUMMARY

In this chapter, we described the general structure of DDS and the OMG specifications included in it to support SIOp. We have also discussed the different advantages of DDS and many of the real world systems that run DDS in both military and civilian organizations. In Chapter IV, we will look at D-DIL environment communication limitations and investigate the implementation of DDS in such an environment with emphasis on SIOp.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ARCHITECTURE DESIGN

In distributed systems, there are problems and challenges to establish network connectivity; unfortunately, there is no one solution for all. The right design is usually a set of trade-offs to keep the balance between resource usage (i.e., memory, bandwidth), network requirements, and the application's constraints. Consequently, the proposed architecture will present a trade-off between a design based on DDS specifications and selected QoS controls to meet the needs of connectivity and SIOp, when deployed for C2 systems operating in D-DIL environment.

Several factors influenced our design but the most important ones to mention are the communication limitations for a disadvantageous network presented in a D-DIL environment and the requirement for semantic interoperability within the subsystems. In the next section, we will describe and evaluate those different factors and investigate the different solutions proposed by using DDS.

A. DDS DESIGN AND DISADVANTAGED NETWORKS

In D-DIL networks, connection is often intermittent, bandwidth is limited, and more challenges could make the communication hard and complex. DDS has addressed those challenges to mitigate them [24]. The evaluation of such an environment forces the researchers to consider many data exchange factors. In the following, those factors and DDS mitigations will be described.

1. Data Compactness

It is important to have an efficient data representation when the available network bandwidth is limited. Smaller payloads take less time to send, favoring the best use of transport layer and decreasing chances to drop messages. To have a successful transmission with SIOp, the data design must result in compact data representation without omitting the semantic interpretation of that data. Some of the current systems adopted XML for data representation, but XML payloads can be large [27].

For the data representation in our design, XML presents a solution as it is supported by DDS but this solution is not efficient in D-DIL networks. Thus, the Common Data Representation (CDR) standard will be adopted in our DDS architecture.

CDR is a compact and efficient binary representation [27] introduced by OMG as a DDS extension. Unlike XML, it minimizes CPU and bandwidth usage. CDR is already implanted in the OMG standards CORBA and the RTPS interoperability protocol. The downside side of using CDR is that analyzing tools need to be added since this is not a human readable representation like XML. Some other solution could be investigated for data compactness solution such as the Efficient XML Interchange (EXI) which is a W3C standard that compresses XML by at least one order of magnitude, without loss of information.

2. Network Recovery

The network protocol must not only use the bandwidth efficiently but also recover quickly from disconnections and losses. While some transport protocols like TCP provide an excellent choice for connectivity, they are most likely to lose performance when it comes to changing network environment. TCP is a connection oriented protocol that needs synchronization and works to establish a guaranteed delivery which can result in large delays, congestion, and very slow recovery in the D-DIL network.

The RTPS wire protocol will be used in our design for data exchange between different entities. This protocol can be layered on top of TCP or UDP [28]. Having RTPS running on top of UDP as best-effort will offer better response to changing network conditions. The UDP used with RTPS would be the design pick for fast and efficient transmission.

3. Limited Resource

In a denied environment, connectivity can be interrupted for an extended period–of-time. Then, the messages sent when the link was down need to be stored and forwarded later on. Here we have two issues. First, the resources needed to manage and store those messages can grow dramatically with the number of nodes/users connected.

Second, after recovery, replaying those messages will be time consuming and can result in network congestion at some point depending on data rates and available bandwidth.

These two problems are addressed by the data-centric message design that allows caching only by a bounded number of messages [24]. DDS uses a data centric design to eliminate dependencies between messages and reduce resource and bandwidth requirements. DDS also supports that design by integrating the RTPS/UDP protocol stack with a variety of QoS policies [29].

4. Graceful Degradation

In networks with large bandwidth and huge resources, message delivery is guaranteed using different technologies and protocols. However, in the case of intermittent network links, it is not possible to deliver every message. Some messaging middleware chooses to drop all messages until having full connectivity back. Other middleware has opted for graceful degradation by delivering as much data as possible. Messages will be dropped or delivered depending on the requirements of the applications such as, time line, updates, and priority.

A data-centric message design connects decoupled applications that eliminate messages' dependencies. The QoS policies supported by DDS such as durability, reliability and ownership [29], enable users/applications to specify the message delivery requirements per stream. This design allows determining when it is appropriate to drop or keep messages in order to meet both the connectivity and interoperability requirements.

B. DDS DESIGN AND SEMANTIC INTEROPERABILITY

As discussed earlier, the SIOp issue presents the ability of different nodes in the network to exchange data efficiently, while preserving the meanings of the exchanged messages regardless of the hardware and software behind those nodes. SIOp issues will be addressed in two different ways. Primarily, at the architecture level, the use of DCPS design enables decoupled applications to exchange information in seamless interoperability. The DCPS design, presented by DDS as MDA, generates data models from the data provided by different applications or participants. That generation takes

place using standard OMG modeling languages like UML and IDL which are designed to preserve semantics. Finally, data entities, which are identified as DDS topics, are generated and implemented using standard programming languages like C++ and Java. Those topics represent the data objects that applications publish or subscribe to and they are maintained and updated only by the middleware (see example in Appendix).

Furthermore, SIOp issues are addressed in the proposed design by implementing the RTPS wire protocol and the DDS API interfaces. The latter enables applications to interact with the DDS global data space by publishing and receiving information immediately using their original data types and data representations. The application programmer is not required to deal with DDS generated topics or to have broad knowledge about the DDS specifications [30]. Shared data between applications are usually common types (i.e., position, weight, temperature), but they may be represented in different units and formats. For example, a location can be presented using different coordinate systems. The DDS API is responsible for discovering the semantics of each data type and providing conversion rules to translate data types between applications back and forth. The API also enables QoS management because it provides direct and dynamic configuration of DDS QoS parameters through configuration files [30].

Another important implementation to support SIOp is the RTPS wire protocol [28]. Although it is used primarily for exchanging messages between nodes, it is also designed to guarantee interoperability between different vendors of DDS implementation without the need for additional configuration. RTPS wire protocol gives the ability to add extensions (i.e., supporting new classes and data types without changing the system design).

C. PROPOSED ARCHITECTURE

Many DDS modular architecture designs were proposed before for different purposes such as mobile systems [31], industrial automation [21], and proxy-based distributed systems [20]. The proposed architecture, shown in Figure 7, is designed for D-DIL environment with SIOp emphasis, and is composed of several modules: the DCPS

API interface, the local storage, the RTPS engine, the RTPS wire protocol, the Controller and the Discovery Protocol (DP).

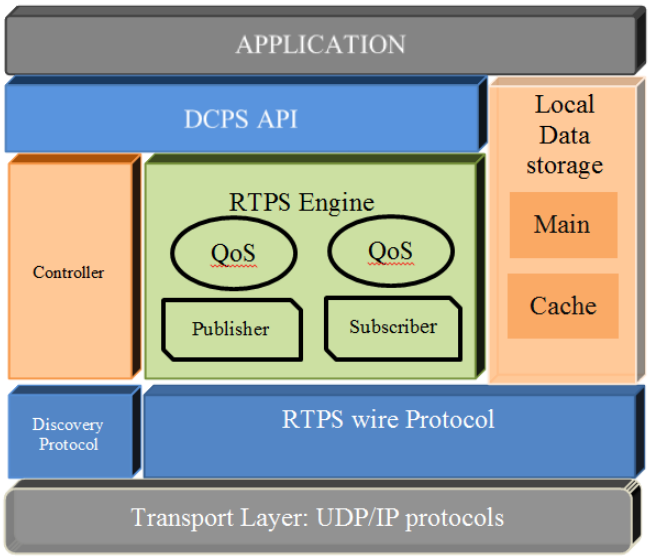


Figure 7. DDS design for D-DIL networks

1. The Application Layer

The application layer on the top represents the original application that connects to the domain through DDS; the information published by the application is conveyed by the DCPS API interface to the DCPS module represented by the RTPS engine and the RTPS Wire Protocol. The example in the appendix describes how the information is modeled and processed from the node and then topics are created and published. In that example, simple RADAR is connected to the network, the information provided was modeled using UML and XML, and then topics are described and programmed via C++.

2. The DCPS API

The DCPS API interface along with the DCPS architecture is the fundamental base to introduce SIOp into our system. The applications communicate by exchanging discoverable data objects also known as Topics which are described using a common data model. A number of semantic modeling standards such as UML, XML, and IDL are used within DDS to model and define those objects. The DCPS API supports many

programming languages and several standards such as C++, Java, and HTTP which ease the integration of application and nodes joining the system.

3. The RTPS Engine

The RTPS engine consists of a number of publishers and subscribers and their QoS controllers. Each publisher is composed from the publisher itself and a Data Writer. Each subscriber consists of a subscribing element and a Data Reader.

The publication takes place when information and updates by the application are converted into Topics and Topic updates which are sent to the local data space through the RTPS engine. The subscription, on the other hand, consists of receiving Topic updates through the RTPS block and saving it in the local data space.

4. The RTPS Wire Protocol

The RTPS wire protocol (RTPS-WP) is the layer of actual data communication. It is responsible for exchanging data messages over the transport layer. The data from Data Writers is propagated by RTPS-WP and saved in the storage. It also notifies Data Readers that Topic information and updates are available. This protocol was designed to run over multicast and connectionless best-effort transmission with many features including:

- Support DCPS for fault tolerance.
- Ability to add new services without giving away compatibility and interoperability.
- Support DDS QoS policies.
- Scalability for larger networks.

5. The Controller

The controller is responsible for detecting the QoS requirements defined by the application layer and translate that to the set of QoS controls and measures that should be considered by Data Writers and Data Readers of each publisher and subscriber. Since those requirements could change over time, the controller has to detect the policies and

status of the publishing application and notify the DDS components (i.e., Data Writers and Data Readers) so they can adjust their QoS controls.

6. The Discovery Protocol

The DP is used to establish communication between publishers and subscribers. It identifies the presence and absence of endpoints, which are critical for the transparent plug-and-play of nodes/applications. The discovery module supports both the Participant DP and the Endpoint DP that work together to identify the existence of entities (Participants) in the domain and their endpoints represented by Data Readers and Data Writers.

D. DEPLOYMENT SCENARIO

The scenario that we created to deploy our DDS architecture consists of a few nodes connected through a data bus. Each node presents the same structure and elements presented in Section C. The goal of this scenario design is to explain how the rich set of QoS policies can be used to support the DDS functionalities in any specific network environment. In our case, the network is operated in a denied environment with just a few nodes; however, the number of nodes can be augmented enormously without affecting DDS performances.

1. Architecture of the Distributed System

The architecture of our distributed system shown in Figure 8 includes three sensors, one workstation, and one Operation Center (OC). It is possible to add several sensors and workstations and any other component to the data bus without damaging the overall functionality of the system thanks to the scalability of DDS, but only a few nodes are presented to illustrate the purpose of this research. Figure 8 depicts an example of an application for the proposed architecture: sensors could symbolize simple components like GPS, temperature sensors, or complex ones such as RADARs. The deployment scenario is not specific to one system; it is rather general and is described by the following:

- Sensor 1: publish Topic A
- Sensor 2: publish Topic B
- Sensor 3: also publish Topic B
- Workstation: subscribe to Topic B, published Topic C
- Operation Center: subscribe to Topics A and C

The network module allows delivering the data from sensors to workstation and OC, and most data will be stored into the remote OC. Data from sensor 1 and workstation should be reliably delivered to OC, data from sensor 2 and from sensor 3 should be delivered to the workstation as soon as possible (best-effort).

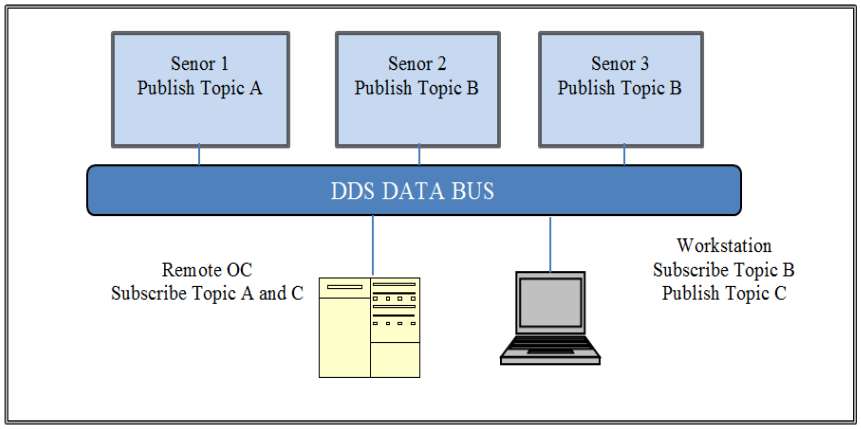


Figure 8. Architecture of the distributed system

2. QoS Settings

The QoS is a set of controls that shape a service behavior. In DDS, QoS are prescribed per data flow and are used to apply fine control of communication between connected nodes. The large set of QoS policies integrated in DDS offers solutions for many technical problems as mentioned in [29], because it enables DDS usage flexibility in different environments and systems. For each publisher-subscriber pair an agreement of QoS is established independently to meet their policies. This unique aspect of DDS gives applications the ability to meet the needs for flexible and complex data transmission. Describing all the QoS policies offered by DDS is beyond the scope of this research. Instead, only a general description will be given. Table 2 depicts the different

QoS policies and their applicability. Those policies are divided into seven different groups:

- Availability: this group of QoS policies controls the availability and expiration time (validity) of published data.
- Delivery: this group controls how data is delivered (reliability, deadlines) and defines the privileges for different Data Writers and Data Readers.
- Infrastructure: this group allows controlling and optimizing the resources used for data distribution.
- Presentation: this group controls how related data samples are presented to subscribers.
- User: this group controls the DDS data needed by each application. These data are typically used by the application to discover the participants in the global data space.
- Redundancy: this group allows information redundancy through multiple Data Writers and Data Readers.
- Transport: this group controls latency and priority for packet dropping and retransmission.

Data Specification	**QoS Policies**
Availability	Durabilty, History, Data Life cycle, Lifespan
Delivery	Reliability, Time Based Filter, Deadline
Infrastructure	Entity Factory, Resource Limits
Presentation	Partition, Presentation, Destination Order
User	User, Topic, Group
Redundancy	Ownership, Ownership Strength, Liveliness
Transport	Latency Badget, Priority

Table 2. QoS policies offered by DDS

Below, we describe selected QoS policies that can have a deep impact on the proposed architectural design.

a. Reliability

To support frequent disconnections in D-DIL networks, we have set specific values for Reliability, Durability, and History QoS policies. Reliability QoS describes the type of data delivery. There are two values: RELIABLE or BEST-EFFORT. RELIABLE means that the publisher uses reliability mechanisms such as acknowledgment and

caching, to ensure that data samples have reached receivers. This will also set the RTPS protocol to work in reliable mode between different endpoints. When the Reliability QoS is set to BEST-EFFORT, applications would tolerate data losses and transmission failures. Considering the characteristics of our network, BEST-EFFORT would be the finest choice, especially for a large number of sensors. However, the QoS value could be set to Reliable for the few critical nodes when guaranteed delivery is needed.

b. Durability

Durability QoS is very important for nodes that join and leave the system occasionally, because it enables them to get the past samples of data available to Data Readers. As described in Figure 9, there are three different ways for DDS to set the Durability QoS controls.

- PERSISTENT: means that data samples will be put in permanent storage and made available for subscribers joining the system in any given time.
- TRANSIENT: data samples are saved in local storage.
- VOLATILE: no history saved, no data instances were stored.

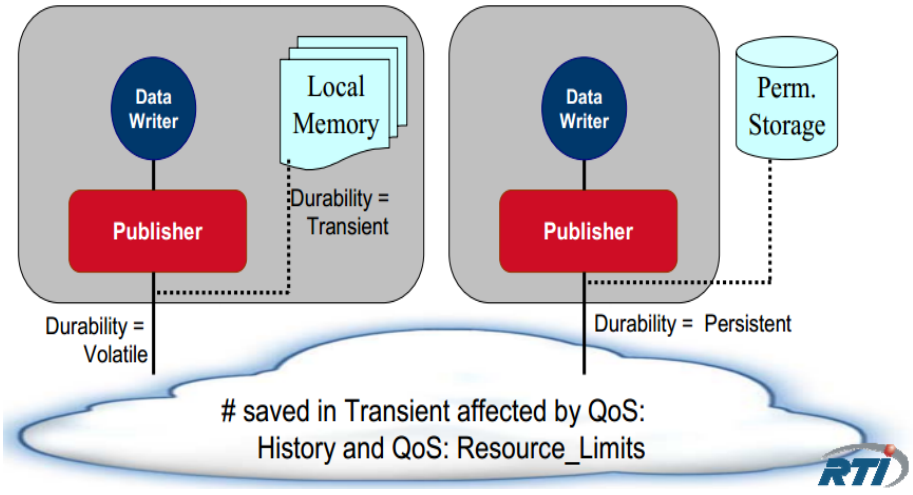


Figure 9. Durability QoS Policy (from [29])

In the previously described deployment scenario, the only entity that could set the durability QoS to PERSISTENT is the OC, which is usually connected to any data base with a permanent large storage. However, for the other entities the TRANSIENT value is

set for workstation and sensor 1 and the VOLATILE value for the other sensors since they typically have little to no storage capacities. For reliable delivery, the Durability QoS cannot be set to VOLATILE since data samples need to be cached for retransmission or stored to update newly connected nodes.

c. History

History QoS policy, described in Figure 10, is strongly tied to the Reliability and Durability QoS policies. It determines how much data is kept by Data Writers and Data Readers when sending and receiving updates. Data is stored locally in the cache memory which is controlled by the History QoS settings. When this policy is configured for KEEP_ALL, then Data Writers and Data Readers cache all the data instances before sending or after receiving them. When History is configured for KEEP_LAST N, then the endpoints will keep only the last N data samples in the buffer. In this case, when saving only the last N data samples, the History QoS policy is tuning the Reliability QoS policy by allowing reduced level of reliability. We recommend the KEEP_LAST N setting for most of the nodes in the network since these nodes usually have very limited memory.

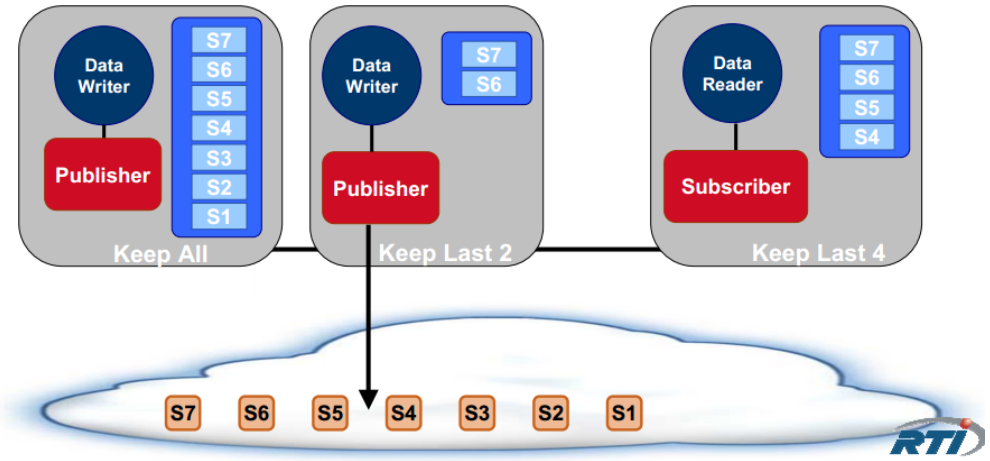


Figure 10. History QoS Policy (from [29])

d. Ownership

The ownership QoS indicates whether a Topic can be updated by one or many Data Writers. The Ownership QoS policy can be configured in two different ways as prescribed in Figure 11. If the publishing application is configured for Exclusive Ownership for a given Topic, then only its Data Writer can send related data samples at a time even if multiple Data Writers are trying to update that topic. In this case, the OwnerShipStrength QoS policy is used to determine the priority for each Data Writer. This QoS policy could be useful at the operational level because it enables fast and good replacement for take-over or fail-over. As an example, consider Topic A where publishing commander's orders to soldiers in the field are typically updated by a field officer. However, if a commander of a higher level decided to take over, he can just join the network and update the same topic by setting the Ownership QoS to Exclusive and the OwnerShipStrength to a higher priority level.

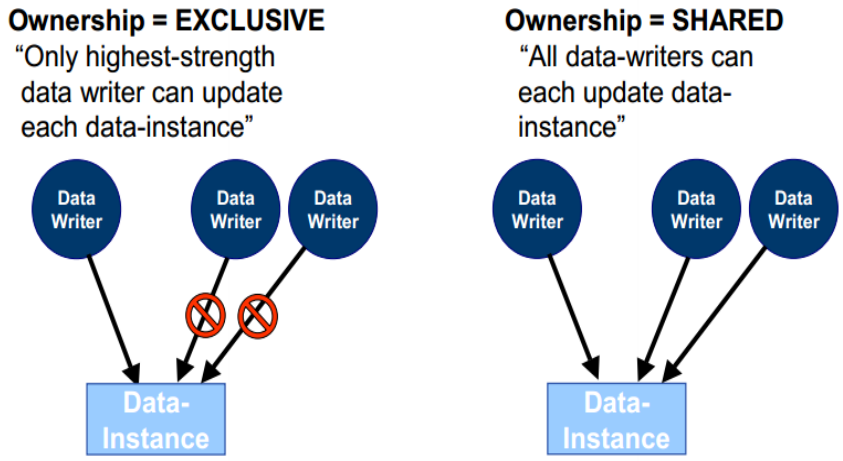


Figure 11. Ownership QoS Policy (from [29])

The Ownership QoS policy can be also set to the value "SHARED." In this case, multiple Data Writers can publish data samples related to the same topic at the same time. For example, we can have two radars sending updates to the same screening device. The Topic can be the graphical interface itself and the two radars sharing ownership for that Topic enable them to send data instances at the same time. In our case scenario, we have

only sensors 2 and 3 that are updating the same topic B and we chose the SHARED setting for Ownership to prove that we can easily have information redundancy to support connectivity and we can also use the Ownership policy to have many sensors sending updates about the same topic without interferences or affecting the overall performance of the system.

e. OwnershipStrength

The OwnershipStrength QoS policy specifies which Data Writer has higher priority to send updates to Data Readers for certain Topics (See Figure 12). This QoS has numerical values to precisely denote the priority when Ownership is exclusive and there are multiple Data Writers updating the same instances.

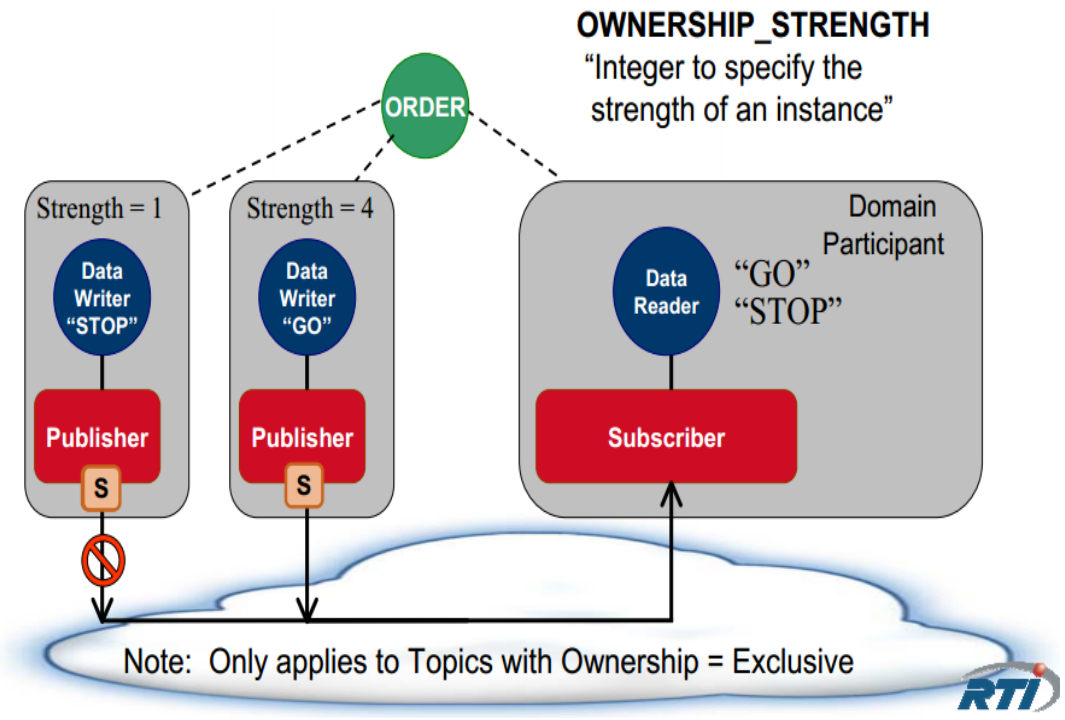


Figure 12. OwnershipStrength QoS Policy (from [29])

f. ContentFilteredTopic

Subscribing applications use the ContentFilteredTopic when they need only a subset of the data published for a given Topic. The filter content is defined by SQL-like

statements and applied to the Data Reader to filter the available data on the related Topic. This QoS policy is useful when it comes to optimize storage performance and reduce data processing resources since only the needed information is kept and processed. The ContentFilteredTopic QoS policy also enables application to design filters for semantic data filtering. In other words, if part or all of the received data does not make sense for the application, then it can be rejected based on the constraints defined in the filter.

To conclude based on the requirements of our system and the different policies explained previously, the brief view of QoS design is described in Table 3.

Node	**Topic**	**QoS Policy**	**Value**
Sensor 1	A	Reliability	RELIABLE
		Durability	TRANSIENT
		History	KEEP_LAST 20
		Ownership	EXCLUSIVE
		OwnershipStrenght	5
Sensor 2	B	Reliability	BEST_EFFORT
		Ownership	SHARED
Sensor 3	B	Reliability	BEST_EFFORT
		Ownership	SHARED
Workstation	B	Durability	TRANSIENT
		History	KEEP_LAST 40
	C	Reliability	RELIABLE
		Durability	TRANSIENT
		History	KEEP_LAST 40

Node	Topic	QoS Policy	Value
		Ownership	EXCLUSIVE
		OwnershipStrenght	1
Operation Center	A	Durability	PERSISTENT
		History	KEEP_ALL
	C	Durability	PERSISTENT
		History	KEEP_ALL

Table 3. QoS settings of the distributed system

E. CONCLUSION

In this chapter, the DDS architecture design and the deployment scenario with the QoS provision presents our attempt to address the challenges of connectivity and semantic interoperability in IT/C2 systems operating in D-DIL environment. The use of distributed networks based on the DDS publish-subscribe infrastructure is certainly efficient when it comes to system integration. It also enables interoperability through open standardization and flexibility through scalable and configurable design. Finally, it promotes the use of semantic interoperability, which is a key building block for advanced command and control.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND RECOMMENDATIONS

The DDS design suggested in this research as well as the deployment scenario, the limitations and the problems are summarized in this last chapter. Some eventual leads for future work will be investigated.

A. SUMMARY

This research was divided into three different parts. The first part introduced the different standards and specifications used to implement SIOp leveraging the work done by the OMG and W3C standards groups. The second part provided a detailed description of the DDS open standard: architecture, functionalities, advantages, and some applications in real-world systems, and then compared DDS to web services as two middleware technologies used to solve integration problems. The last part contains our suggested DDS design to meet the communication and interoperability requirements in a denied environment (D-DIL). That design was implemented in a general scenario to exemplify the tune-up of the system controls offered by the QoS provisions.

The DDS design was based on the specifications described by the OMG and customized to the network environment limitations including limited resources, connection degradation and recovery, and data representation. The different protocols and interfaces used were related to the integration of the different nodes of the network and contributed to solve the SIOp issues within the given system. Finally, the configuration of the QoS policies was dedicated to make the system work more efficiently and accurately. Our overall idea is innovative, and while we have not implemented any scientific testing, the design seems relatively successful in terms of simplicity and feasibility.

B. RECOMMENDATIONS AND FUTURE WORK

This research could be viewed as the first step to develop a middleware for system integration in D-DIL environment. Many commercial solutions can be used; however, to adapt one of them to a specific system can be hard and complex. Hence, we suggest the following:

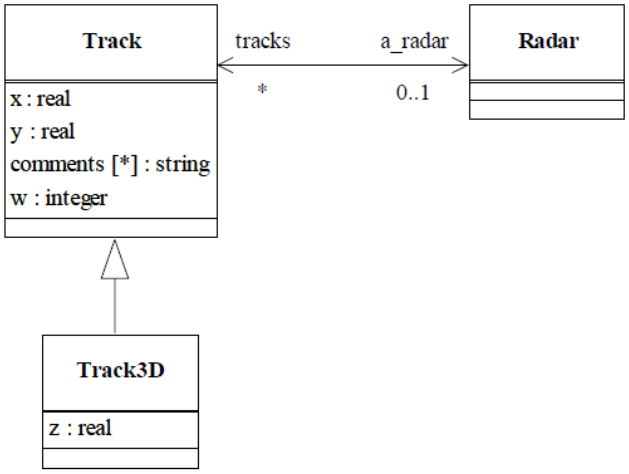
- Develop and code a basic DDS solution for denied environment. Some of the offered solution can be modified to include a DDS minimum profile.
- Use some tools to help design and code interfaces for different systems. For example, the DDS code generator built by the Milsoft Company.
- Experiment and test for better development using tools such as DDS Spy by Milsoft Company and Recording and Replay for DDS by the Real Time Innovation Company.
- Further DDS related issues could be investigated such as mobility and security.
- Model the D-DIL DDS environment to determine our weak points and how much it scales given certain operational plans.
- Conduct thesis research on the efficacy of the Efficient XML Interface (EXI) W3C standard as it would apply to this scenario.

APPENDIX. DATA MODELING IN DDS

This example is taken from the DDS specification published by the OMG group in [19]. Its purpose is to show how data is processed by DDS from simple UML diagram to the final coding for Topics that are published in the global data space.

A. UML MODEL

The following UML diagram describes a very simple application model with three classes [19].



UML class diagram of the example (from [19])

B. IDL MODEL DESCRIPTION

Based on the IDL model [19], the model description (IDL provided by the application developer) could be:

#include “dlrl.idl”

valuetype stringStrMap; // StrMap<string>

valuetype TrackList; // List<Track>

valuetype Radar;

```
valuetype Track : DLRL::ObjectRoot {
public double x;
public double y;
public stringStrMap comments;
public long w;
public Radar a_radar;
};
valuetype Track3D : Track {
public double z;
};
valuetype Radar : DLRL::ObjectRoot {
public TrackList tracks;
 };
```

C. XML MODEL TAGS

The UML tags [19] to drive the generation process could then be:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Dlrl SYSTEM "dlrl.dtd">
<Dlrl name="example">
<templateDef name="StringStrMap" pattern="StrMap" itemType="string"/>
 <templateDef name="TrackList" pattern="List" itemType="Track"/>
<classMapping name="Track">
<mainTopic name="TRACK-TOPIC">
<keyDescription content="FullOid">
<keyField>CLASS</keyField>
```

<keyField>OID</keyField>

</keyDescription>

</mainTopic>

<monoAttribute name="x">

<valueField>X</valueField>

</monoAttribute>

<monoAttribute name="y">

<placeTopic name="Y_TOPIC">

<keyDescription content="FullOid">

<keyField>CLASS</keyField>

<keyField>OID</keyField>

</keyDescription>

</placeTopic>

<valueField>Y</valueField>

</monoAttribute>

<multiAttribute name="comments">

<multiPlaceTopic name="COMMENTS-TOPIC" indexField="INDEX">

<keyDescription content="FullOid">

<keyField>CLASS</keyField>

<keyField>OID</keyField>

</keyDescription>

</multiPlaceTopic>

<valueField>COMMENT</valueField>

</multiAttribute>

<monoRelation name=“a_radar”>

<keyDescription content=“SimpleOid”>

<keyField>RADAR_OID</keyField>

</keyDescription>

</monoRelation>

<local name=“w”/>

</classMapping>

<classMapping name=“Track3D”>

<mainTopic name=“TRACK-TOPIC”>

<keyDescription content=“FullOid”>

<keyField>CLASS</keyField>

<keyField>OID</keyField>

</keyDescription>

</mainTopic>

<extensionTopic name=“TRACK3D-TOPIC”>

<keyDescription content=“FullOid”>

<keyField>CLASS</keyField>

<keyField>OID</keyField>

</keyDescription>

</extensionTopic>

<monoAttribute name=“z”>

<valueField>Z</valueField>

</monoAttribute>

</classMapping>

<classMapping name="Radar">

<mainTopic name="RADAR-TOPIC">

<keyDescription content="SimpleOid">

<keyField>OID</keyField>

</keyDescription>

</mainTopic>

<multiRelation name="tracks">

<multiPlaceTopic name="RADARTRACKS-TOPIC" indexField="INDEX">

<keyDescription content="SimpleOid">

<keyField>RADAR-OID</keyField>

</keyDescription>

</multiPlaceTopic>

<keyDescription content="FullOid">

<keyField>TRACK-CLASS</keyField>

<keyField>TRACK-OID</keyField>

</keyDescription>

</multiRelation>

</classMapping>

<associationDef>

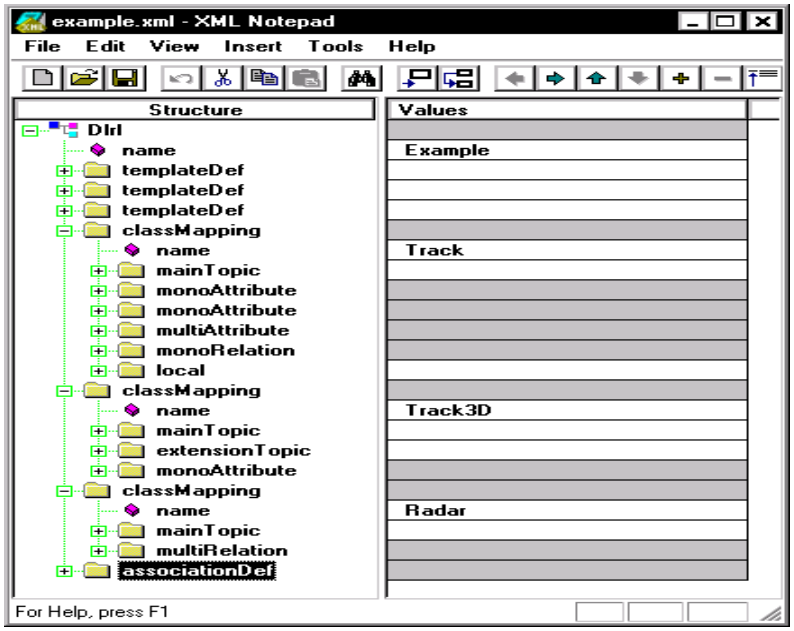
<relation class="Track" attribute="a_radar"/>

<relation class="Radar" attribute="tracks"/>

</associationDef>

</Dlrl>

It should be noted that XML is not suitable for manual editing [19]; therefore, the file seems much more complicated than it actually is. It seems much simpler when viewed through an XML editor, as Figure 14 illustrates.



XML Editor Illustration (From [19])

Only the three templateDef, the associationDef, and the local17 tags are mandatory in all cases. The ClassMapping tags are only required if a deviation is wanted from the default mapping. In case no deviation is wanted from the default mapping, the XML description can be restricted to the following minimum:

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE Dlrl SYSTEM "dlrl.dtd">

<Dlrl name="Example">

<templateDef name="stringStrMap" pattern="StrMap" itemType="string"/>

<templateDef name="TrackList" pattern="List" itemType="Track"/>

<classMapping name="Track">

<local name="w"/>

</classMapping>

<associationDef>

<relation class="Track" attribute="a_radar"/>

<relation class="Radar" attribute="tracks"/>

</associationDef>

</Dlrl>

A following step could be to define UML 'tags'18 and to generate those files based on the UML model. However, this is far beyond the scope of this specification.

D. UNDERLYING DCPS DATA MODEL

This mapping description assumes that the underlying DCPS data model is made of five topics with their fields as described in Table 4.

TRACK-TOPIC	Topic to store all Track objects (including the derived classes) – as well as the embedded attributes/relations defined on Track.
CLASS	Field to store the class part of the object reference.
OID	Field to store the oid part of the object reference.
X	Field to store the value of the attribute x.
RADAR-OID	Field to store the relation a_radar.

Y-TOPIC	Topic to store Track::y, outside Track's main topic.
CLASS	Field to store the class part of the object reference.
OID	Field to store the oid part of the object reference.
Y	Field to store the value of the attribute y.

COMMENTS-TOPIC	Topic to store Track::comments (required as it is a collection).
CLASS	Field to store the class part of the owning object reference (here a Track).
OID	Field to store the oid part of the owning object reference (here a Track).
INDEX	Field to store the index part in the collection
COMMENT	Field to store one element of the attribute comments.

TRACK3D-TOPIC	Topic to store the embedded attributes/relations added on Track3D (here only z).
CLASS	Field to store the class part of the object reference.
OID	Field to store the oid part of the object reference.
Z	Field to store the value of the attribute z.

RADARTRACKS-TOPIC	Topic to store Radar::tracks (required, as it is a collection).
RADAR-OID	Field to store the reference to the owning object (here a Radar).
INDEX	Field to store index in the collection.
TRACK-CLASS	Field to store the class part of a reference to an item in the collection (here a Track).
TRACK-OID	Field to store the oid part of a reference to an item in the collection (here a Track).

Topics in the DCPS model (From [19])

Note that references to Track objects (including derived Track3D) must provision a field for the class indication, while references to Radar objects do not, for the Radar class has no subclasses and does not share its main Topic.

E. CODE EXAMPLE

The following text is a very simple, non-fully running, C++ example just to give the flavor of how objects can be created, modified, and then published.

```
DDS::DomainParticipant_var dp;

DLRL::CacheFactory_var cf

/*

  * Init phase

*/ DLRL::Cache_var c = cf->create_cache (WRITE_ONLY, dp);

RadarHome_var rh;

TrackHome_var th;

Track3DHome_var t3dh;

c->register_home (rh);
```

```
c->register_home (th);
c->register_home (t3dh);
c->register_all_for_pubsub();
// some QoS settings if needed
c->enable_all_for_pubsub();
/*
 * Creation, modifications and publication
*/
Radar_var r1 = rh->create_object(c);
Track_var t1 = th->create-object (c);
Track3D_var t2 = t3dh->create-object (c);
t1->w(12);                  // setting of a pure local attribute
t1->x(1000.0);                    // some DLRL attributes settings
t1->y(2000.0);
t2->a_radar->put(r1);             // modifies r1->tracks accordingly
t2->x(1000.0);
t2->y(2000.0);
t2->z(3000.0);
t2->a_radar->put(r1);   // modifies r1->tracks accordingly
c->write();                 // all modifications are published
};
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] L. R. Trebaol, (2013, July 29), *The Path to Semantic Interoperability*. [Online]. Available: http://blogs.rti.com/2013/07/29/the-path-to-semantic-interoperability-part-2/

[2] M. Patel et al., "Semantic interoperability in Digital Library Systems,." UKOLN, University of Bath, UK. Rep. IST-2002-2.3.1.12, 2005.

[3] G. Hunt. "Interoperable Open Architecture-Toward Minimizing UAS Acquisition Cost," UAS Conference. Paris, France, 2011.

[4] UCS UAS Control Segment. (2011). *The Data Distribution Service: Reducing Cost through Agile Integration*. [Online]. Available: http://www.twinoakscomputing.com/wp/DDS_Exec_Brief_v20l-public.pdf

[5] Object Management Group, (2011, November 14), *Semantic Information Modeling for Federation (SIMF)*. [Online]. Available: http://www.omgwiki.org/architecture-ecosystem/doku.php?id=semantic_information_modeling_for_federation_rfp

[6] S. Schneider and B. Farabaugh, (2009, April 29). *Is DDS for You?* [Online]. Available: https://www.rti.com/mk/DDS.html

[7] Object Management Group, (2013, June 01), *Meta Object Facility (MOF) Core Specification (version 2.4.1)*. [Online]. Available: http://www.omg.org/spec/MOF/2.4.1/

[8] Object Management Group. (2013, April 02). *MOF Support For Semantic Structure*. [Online]. Available: http://www.omg.org/spec/SMOF/

[9] Object Management Group. (2011, August 05). *Unified Modeling Language (version 2.4.1)*. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/

[10] Object Management Group. (2012, January 01). *OMG Object Constraint Language (version 2.3.1). [Online]*. Available: http://www.omg.org/spec/OCL/2.3.1/

[11] Object Management Group. (2011, February 02). *Semantics of a Foundational Subset for Executable UML Models (version 1.0)*. [Online]. Available: http://www.omg.org/spec/FUML/1.0/

[12] Object Management Group. (2010, September 15). *Model Driven Message Interoperability (version 1.0)*. [Online]. Available: http://www.omg.org/spec/MDMI/1.0/

[13] World Wide Web Consortium. *W3C Mission*. [Online]. Available: http://www.w3.org/Consortium/mission.html

[14] P. Hitzler et al. (2009, October 27). *OWL 2 Web Ontology Language*. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

[15] S. Gao et al. (2012, April 05). *W3C XML Schema Definition Language (XSD 1.1*. [Online]. Available: http://www.w3.org/TR/xmlschema11-1/

[16] World Wide Web Consortium. (2013, June 27). *Linked Data Glossary*.. [Online]. Available: http://www.w3.org/TR/ld-glossary/

[17] World Wide Web Consortium. (2013). *Linked Data*. [Online]. Available: http://www.w3.org/standards/semanticweb/data

[18] J. Heflin and J. Hendler. *Semantic Interoperability on the Web*. University of Maryland, Adelphi, MD. Rep. 0704–0188. 2000.

[19] Object Management Group, (2007, January 01). *Data Distribution Service for Real-Time Systems (version 1.2)*. [Online]. Available: http://www.omg.org/spec/DDS/1.2

[20] K. J. Kwon et al., "A proxy-based approach for mobility support in the DDS system," in *6th IEEE International Conference on Industrial Informatics*. Daejeon, South Korea. 2008, pp.1200–1205.

[21] J. Yang et al., "Data Distribution Service for industrial automation," in *17th Conference on Emerging Technologies & Factory Automation*. Krakow, Poland. 2012 IEEE. doi: 10.1109/ETFA.2012.6489544

[22] M. Fang, et al., "Design of real-time distributed system using DDS," in *IEEE International Conference on Automatic Control and Artificial Intelligence*. Xiamen, China. 2012, pp. 882–885.

[23] Real Time Innovation. *DDS standard*.. [Online]. Available: http://www.rti.com/products/dds/omg-dds-standard.html

[24] Object Management Group. (2011, August 01). *Data Distribution Service Brief*. [Online]. Available: http://portals.omg.org/dds/content/document/data-distribution-service-dds-brief

[25] R. Townsen. "Ship Self Defence System Mk 2 and Data Distribution Standard," Raytheon Integrated Defense Systems Co., Waltham, MA. Rep. ltr 5720/00DT September 2006.

[26] A. M. Kulkarni and P. B. Rao, “Comparative study of middleware for C4I systems: Web Services vis-à-vis Data Distribution Service,” in *IEEE 2012 International Conference on Recent Advances in Computing and Software Systems*. Chennai, India. 2012, pp. 305–310.

[27] Object Management Group. (2011, March 01). *Extensible and Dynamic Topics for DDS*. (2011). [Online]. Available: http://portals.omg.org/dds/sites/default/files/x-types_ptc_11-03-11.pdf

[28] Real Time Innovation. (2002, February) Real-*Time Publish-Subscribe Wire Protocol Specification*. [Online]. Available: http://orte.sourceforge.net/

[29] G. A. Hunt. “DDS–Using QoS to Solve Real-World Problems,” in *OMG Real-Time and Embedded Workshop*. Arlington, VA. July 2012.

[30] G. Baptista et al., “Middleware Supporting Net-Ready Applications for Enhancing Situational Awareness on Rotorcraft,” in *9th International Conference on Autonomic and Autonomous Systems*. Lisbon, Portugal. 2013, pp. 46–52.

[31] K. J. Kwon et al. “DDSS: A Communication Middleware based on the DDS for Mobile and Pervasive Systems,” in *10th International Conference on Advanced Communication Technology*. Gangwon-Do, South Korea. 2008, pp.1364–1369.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California